Model-Based Design for effective HW/SW Co-Design

Alexander Schreiber – Senior Application Engineer
MathWorks, Germany
Agenda

- Model-Based Design of embedded Systems
- Software Implementation and Verification
  - Automatic C/C++ code generation
- Hardware Implementation and Verification
  - Floating- to fixed-point conversion
  - Automatic HDL code generation
  - Code optimization (speed, area, power)
  - Verification on different levels of abstraction (HDL co-simulation, FIL)
- HW/SW Integration on heterogeneous System-on-Chips (SoCs)
  - Execution profiling (SIL, PIL)
  - Generation of target-specific C code
  - Generation of hardware IP interfacing internal system bus
  - Workflow for HW/SW system integration
- Questions & Answers
Things to remember ….

- **Best Practice #1:**  
  Enable collaboration by integrating workflows with Model-Based Design

- **Best Practice #2:**  
  Reduce development time with Automatic Code (C, HDL) generation

- **Best Practice #3:**  
  Reduce verification time with Test Bench Reuse
Model-Based Design of embedded Systems
Demo: PMSM Control

Gigabit Ethernet
ARM Cortex-A9 Processor

- System stimuli
- Voltage rectification
- 3-phase inverter
- Signal measurement
- PMSM motor model

Plant Model

- 220V 60Hz
- Braking chopper
- Three-phase Inverter
- Measures
- Permanent Magnet Synchronous Machine
- Motor

AXI Bus

Programmable Logic

- Field Oriented Control
- Coordinate transforms
- Torque control loop
- Speed control loop
- Vector modulation

HDL Code Generation

- Embedded Coder™
- C Code Generation

HDL Coder™

- HDL Code Generation

Controller

- Group 1
- Signal 1
- Speed
- [label]
Traditional Flow

Requirement Documents
- Difficult to analyze
- Difficult to manage as they change

Paper Specifications
- Easy to misinterpret
- Difficult to integrate with design

Physical Prototypes
- Incomplete and expensive
- Prevents rapid iteration
- No system-level testing

Manual Coding
- Time consuming
- Introduces defects and variance
- Difficult to reuse

Traditional Testing
- Design and integration issues found late
- Difficult to feed insights back into design process
- Traceability

EDA
Electrical Components

Algorithm Design
Embeddable Algorithms

MCAD/MCAE
Mechanical Components

C/C++
Embedded Software

HDL
FPGA/ASIC

RESEARCH

REQUIREMENTS

SPECIFICATIONS

DESIGN

IMPLEMENTATION

INTEGRATION AND TEST
Seamless Development

RESEARCH

REQUIREMENTS

DESIGN

- Environment Models
- Physical Components
- Algorithms

IMPLEMENTATION

- C, C++
- VHDL, Verilog
- SPICE
- MCU
- DSP
- FPGA
- ASIC
- Analog Hardware

INTEGRATION

TEST & VERIFICATION

TEST SYSTEM

Idea

Algorithm

FPGA, DSP

Model-Based Design
Heterogeneous Systems-on-Chip (SoCs)

- Single-chip solution containing
  - SW processors
  - HW co-processors and interfaces
  - Chip-internal system bus
  - Optional: analog IPs

- Challenges
  - HW/SW partitioning
  - System integration
Software Implementation and Verification
Model-Based Design

RESEARCH

REQUIREMENTS

DESIGN

Environment Models

Timing and Control Logic

Digital Models

Analog Models

Algorithms

IMPLEMENTATION

C, C++

VHDL, Verilog

SPICE

MCU

DSP

FPGA

ASIC

Analog Hardware

INTEGRATION

TEST & VERIFICATION

TEST SYSTEM
C-Code Generation Tools

- MATLAB Coder
- Simulink Coder
- Embedded Coder
MATLAB Coder

- Automatic ANSI C code generation from MATLAB
  - Simulink is not required
- A GUI for project generation from MATLAB
- Use configurations to control the generated code
Simulink Coder

- C code generation from Simulink, Stateflow and Simscape
- Hardware-in-the-loop testing (external mode)
- Targeting desktop applications
  - Supports Eclipse IDE
  - Includes Windows OS and Linux OS Target support package
Embedded Coder

- Major consolidation of MathWorks products to provide a high value solution
- Targeting real-time embedded systems
  - Code optimization / customization
  - SIL / PIL
  - Profiling
Hardware Implementation and Verification
Model-Based Design

RESEARCH

REQUIREMENTS

DESIGN

Environment Models

Timing and Control Logic

Digital Models

Analog Models

Algorithms

IMPLEMENTATION

C, C++

VHDL, Verilog

SPICE

MCU, DSP, FPGA, ASIC

Analog Hardware

INTEGRATION

TEST & VERIFICATION

TEST SYSTEM
Separate Views of DSP Implementation

- System Designer
  - Algorithm Design
    - Fixed-Point
    - Timing / Control Logic
    - Architecture Exploration
    - Algorithms / IP
  - System Test Bench
    - Environment Models
    - Analog Models
    - Digital Models
    - Algorithms / IP
  - FPGA Requirements
    - Hardware Specification
    - Test Stimulus

- FPGA Designer
  - RTL Design
    - IP Interfaces
    - HW Architecture
  - Implement Design
    - Synthesis
    - Map
    - Place & Route
  - Verification
    - Behavioral Simulation
    - Functional Simulation
    - Static Timing Analysis
    - Timing Simulation
    - Back Annotation
  - FPGA Hardware
Where do you spend most of your time?

- Simulating designs?
- Creating designs and test benches?
- Analyzing and combining results from multiple tools?
- Exploring implementation ideas and architectures?
- Floating point to fixed-point?
- Writing HW specifications?
- Iterating over designs with the FPGA designer?
- Blaming the FPGA designer?
Where do you spend most of your time?

- Simulating designs and validating against HW specs?
- Creating designs and writing test benches?
- Hardware architecture design?
- Writing interfaces to existing IP?
- Synthesis, Map, PAR cycles?
- Iterating over designs with the system designer?
- Blaming the system designer?
A Few Ways to Reduce Development Time

1. Increase simulation speed
2. Simplify design entry, system test harness creation, and exploration
3. Automate RTL design & verification to have shorter iteration cycles
4. Integrate the separate workflows to facilitate collaboration, re-use, and prototyping
Model-Based Design for Implementation

MATLAB® and Simulink®
Algorithm and System Design

Algorithm Design
- Fixed-Point
- Timing / Control Logic
- Architecture Exploration
- Algorithms / IP

System Test Bench
- Environment Models
- Analog Models
- Digital Models
- Algorithms / IP

FPGA Requirements
- Hardware Specification
- Test Stimulus

RTL Design
- IP Interfaces
- Hardware Architecture

Verification
- Behavioral Simulation
- Functional Simulation
- Static Timing Analysis
- Timing Simulation
- Back Annotation

Implement Design
- Synthesis
- Map
- Place & Route

FPGA Hardware
MATLAB® and Simulink®
Algorithm and System Design
Model Refinement for Hardware

Automatic HDL
Code Generation

**Model-Based Design for Implementation**

- **RTL Design**
  - IP Interfaces
  - Hardware Architecture

- **Implement Design**
  - Synthesis
  - Map
  - Place & Route

- **Verification**
  - Behavioral Simulation
  - Functional Simulation
  - Static Timing Analysis
  - Timing Simulation
  - Back Annotation

- **FPGA Hardware**
Model-Based Design for Implementation

MATLAB® and Simulink®
Algorithm and System Design
Model Refinement for Hardware

Automatic HDL Code Generation ➔ HDL Co-Simulation

Behavioral Simulation

Verification
- Behavioral Simulation
- Functional Simulation
- Static Timing Analysis
- Timing Simulation
- Back Annotation

Implement Design
- Synthesis
- Map
- Place & Route

FPGA Hardware
Model-Based Design for Implementation

MATLAB® and Simulink®
Algorithm and System Design
Model Refinement for Hardware

Automatic HDL Code Generation
→ HDL Co-Simulation
→ Behavioral Simulation

Implement Design
- Synthesis
- Map
- Place & Route

Verification
- Functional Simulation
- Static Timing Analysis
- Timing Simulation
- Back Annotation

Implement Design
- Synthesis
- Map
- Place & Route

Verification
- Functional Simulation
- Static Timing Analysis
- Timing Simulation
- Back Annotation

FPGA Hardware
Model-Based Design for Implementation

MATLAB® and Simulink®
Algorithm and System Design
Model Refinement for Hardware

Automatic HDL Code Generation
HDL Co-Simulation
Behavioral Simulation
Back Annotation

Implement Design
Synthesis
Map
Place & Route

Verification
Functional Simulation
Static Timing Analysis
Timing Simulation

FPGA Hardware
FPGA-in-the-Loop

FPGA Hardware
Automatic HDL Code Generation
From Algorithm to Synthesizable RTL

MATLAB® and Simulink®
Algorithm and System Design
Model Refinement for Hardware

Automatic HDL Code Generation

HDL Co-Simulation
Behavioral Simulation

Implement Design
Synthesis
Map
Place & Route

Verification
Functional Simulation
Static Timing Analysis
Timing Simulation

FPGA Hardware
FPGA-in-the-Loop

Back Annotation
Fixed-Point Analysis

Corner Detection

- Convert floating point to **optimized** fixed-point models
  - Automatic tracking of signal range (also intermediate quantities)
  - Word / Fraction lengths recommendation
- Bit-true models in the same environment
Algorithm to HDL Workflows

Simulink to HDL
(with MATLAB and Stateflow)
MATLAB to HDL
Hybrid workflow

VHDL & Verilog
Automatic HDL Code Generation

Automatically generate bit true, cycle accurate HDL code from Simulink, MATLAB and Stateflow

Full bi-directional traceability!!
Simulink Library Support for HDL

HDL Supported Blocks

- 170 blocks supported

- **Core Simulink Blocks**
  - Basic and Array Arithmetic, Look-Up Tables, Signal Routing (Mux/Demux, Delays, Selectors), Logic & Bit Operations, Dual and single port RAMs, FIFOs, CORDICs, Busses

- **Signal Processing Blocks**
  - NCOs, FFTs, Digital Filters (FIR, IIR, Multi-rate, Adaptive), Rate Changes (Up & Down Sample), Statistics (Min/Max)

- **Communications Blocks**
  - Psuedo-random Sequence Generators, Modulators / Demodulators, Interleavers Deinterleavers, Viterbi Decoders
MATLAB & Stateflow for HDL

HDL Supported Blocks

- MATLAB
  - Relevant subset of the MATLAB language for modeling and generating HDL implementations
  - `eml_hdl_design_patterns`: Useful MATLAB Function Block Design Patterns for HDL

- Stateflow
  - Graphical tool for modeling Mealy and Moore Finite State Machines
Integrating Legacy HDL Code
HDL Supported Blocks

- Integrate legacy HDL code in Simulink using black boxes
- Configure the interface to legacy HDL code
- HDL Verifier is a special black box
Automated Mapping to Floating Point HDL
FPGA Vendor Floating Point Libraries

- Support for:
  - Floating Point Altera Megafunctions
  - Xilinx LogiCORE IP Floating Point Operator
  - Singles and Doubles support
HDL Code Optimization
From Algorithm to Optimized RTL

MATLAB® and Simulink®
Algorithm and System Design
Model Refinement for Hardware

Automatic HDL Code Generation

Implement Design
- Synthesis
- Map
- Place & Route

HDL Co-Simulation

Behavioral Simulation

Verification
- Functional Simulation
- Static Timing Analysis
- Timing Simulation

Back Annotation

FPGA Hardware FPGA-in-the-Loop

MATLAB EXPO 2013
How can you easily explore different implementation solutions?
Hardware Design Challenges: Timing Analysis

Finding the critical path in your model can be challenging.
Strategies for Speed Improvement

- Pipelining
  - Input / Output pipeling
  - (Hierarchical) Distributed pipelining
  - Delay Balancing

- Architectural choices, e.g.
  - Linear, tree, cascade
  - Factored-Canonical-Signed-Digit (FCSD)
  - Newton-Raphson Approximation
  - CORDIC
Identifying the critical path

Integrating with P&R Timing Analysis

- Critical Path highlighting:
- Visual representation of critical path in your model
  Easier to identify bottlenecks of your model
Hardware Design Challenges:
Balance Pipeline Registers

- Multiple parallel paths through your model
- High risk to have unmatched latencies
Hardware Design Solution: Distributed Pipelining
Strategies for Area Optimization

- **Goal**
  - Area reduction

- **Means**
  - Time-multiplexed re-use of resources

- **Algorithms**
  - Resource Sharing
    - Re-use of identical operators or atomic subsystems within algorithm
  - Resource Streaming
    - Re-use of vectorized operators or subsystems
Strategies for Power Optimization

Power Dissipation = Static Power + Dynamic Power

- Static Power = Due to transistor leakage current
  - Significant in smaller silicon geometries
- Dynamic Power = $\frac{1}{2}CV^2fA$
  - Function of load capacitance, operating frequency, activity level and voltage swing

Steps To Reduce Power:
- Smaller/Efficient Designs ➔ fixed-point optimization
- Reduce Clock Frequency ➔ gated clocks, multiple clocks
- Control Subsystem Execution ➔ gated clocks
- Low Power Design Libraries/FPGA Devices
Multi-rate Models to Reduce Clock Frequency
Power Optimization

- Cycle accurate simulation and implementation
- Multiple or single clock implementation
Control Subsystem Execution

Power Optimization

Enabled Subsystems

- Modules can be enabled and disabled
Control Subsystem Execution
Power Optimization

Triggered Subsystems
- Modules can be triggered: rising / falling / either edge
HDL Verification
Integrated HDL Verification

MATLAB® and Simulink®
Algorithm and System Design
Model Refinement for Hardware

Automatic HDL Code Generation

HDL Co-Simulation

Behavioral Simulation

Back Annotation

Implement Design

Synthesis
Map
Place & Route

Verification

Functional Simulation
Static Timing Analysis
Timing Simulation

FPGA Hardware
FPGA-in-the-Loop
Stand-Alone HDL Verification

Simulink Test bench

Stimulus

Simulink Design
Targeted to Hardware

Reference Results

Automatically Generated HDL Test Bench

Stimulus

HDL Design

Actual Results
HDL Co-Simulation

- Re-use system level test bench
- Combine analysis in HDL Simulator and MATLAB/Simulink
FPGA-In-the-Loop Verification

- Re-use the MATLAB/Simulink test bench
- Accelerate Verification with FPGA Hardware

MATLAB/ Simulink Testbench

Stimulus

Input stimuli

Response

Output response

HDL Verifier

Connects FPGA HW with the MATLAB environment!
HW/SW Integration on heterogeneous Systems-on-Chip (SoCs)
Targeting Heterogeneous Systems
Partitioning Through Execution Weights

- Hand Code: C, C++, HDL
- IDDE
- Requirements Capture
- System Model: Functional Design, Float to Fixed Point, Plant Model
- Simulation and Analysis
- Execution Weights
- Partitioning and Implementation through Code Generation
- Integration
- Final Design
- C, C++, ASM for MCUs & DSPs
- VHDL, Verilog for ASICs & FPGAs
- IDDE
Software Execution Profiling (SIL, PIL)

- Measures the execution time of the code on the specific target platform during PIL simulations
Target-specific C Code Generation

Exp.: Embedded Coder Support for ARM Cortex-A9
Target-specific C Code Generation
Exp.: Embedded Coder Support for ARM Cortex-A9

- Access on NEON instructions for vector / matrix arithmetic operations
- Leverage processor peripherals efficiently
Targeting Heterogeneous Systems
Partitioning Through Execution Weights

Requirements Capture

System Model
Functional Design
Float to Fixed Point
Plant Model

Hand Code
C, C++, HDL

IDDE

Simulation and Analysis

Partitioning and Implementation
through Code Generation

Execution Weights

C, C++, ASM
for MCUs & DSPs

Integration

C, C++, ASM
for MCUs & DSPs

Final Design

VHDL, Verilog
for ASICs & FPGAs

IDDE

Capture
Simulation and Analysis
Partitioning and Implementation
through Code Generation
Integration
Final Design

MathWorks
High-Level Design Flow

- Fixed-Point Designer (opt.)
- MATLAB Coder
- Simulink Coder
- Embedded Coder

- Simulink
- Simscape
- SimPowerSystems
- Fixed-Point Designer
- MATLAB Coder
- HDL Coder
- HDL Verifier
Summary
Things to remember ....

- **Best Practice #1:**
  Enable collaboration by integrating workflows with Model-Based Design

- **Best Practice #2:**
  Reduce development time with Automatic Code (C, HDL) generation

- **Best Practice #3:**
  Reduce verification time with Test Bench Reuse
Model-Based Design - Basis

RESEARCH
REQUIREMENTS
MATLAB

DESIGN
- Environment Models
- Timing and Control Logic
- Digital Models
- Analog Models
- Algorithms

IMPLEMENTATION
- C, C++
- VHDL, Verilog
- SPICE
- MCU
- DSP
- FPGA
- ASIC
- Analog Hardware

INTEGRATION

TEST & VERIFICATION

TEST SYSTEM
Model-Based Design - Basis

- **RESEARCH**
- **REQUIREMENTS**
- **DESIGN**
  - Environment Models
  - Timing and Control Logic
  - Digital Models
  - Analog Models
  - Algorithms
- **IMPLEMENTATION**
  - C, C++
  - VHDL, Verilog
  - SPICE
  - MCU
  - DSP
  - FPGA
  - ASIC
  - Analog Hardware
- **TEST & VERIFICATION**
- **TEST SYSTEM**
- **MATLAB**
- **Simulink**
Model-Based Design - Basis

- **RESEARCH**
- **REQUIREMENTS**
- **DESIGN**
  - Environment Models
  - Timing and Control Logic
  - Digital Models
  - Analog Models
  - Algorithms
- **IMPLEMENTATION**
  - C, C++
  - VHDL, Verilog
  - SPICE
  - MCU
  - DSP
  - FPGA
  - ASIC
  - Analog Hardware
- **TEST & VERIFICATION**
  - MATLAB
  - Simulink
  - Stateflow
- **INTEGRATION**
Model-Based Design - Basis

RESEARCH → REQUIREMENTS → DESIGN → IMPLEMENTATION → INTEGRATION → TEST & VERIFICATION

DESIGN:
- Environment Models
- Timing and Control Logic
- Digital Models
- Analog Models
- Algorithms

IMPLEMENTATION:
- C, C++
- VHDL, Verilog
- SPICE
- MCU
- DSP
- FPGA
- ASIC
- Analog Hardware

TEST SYSTEM:
- MATLAB
- Simulink
- Stateflow
- Fixed-Point Designer
Model-Based Design – HDL Code Generation

RESEARCH

REQUIREMENTS

DESIGN

Environment Models

Timing and Control Logic

Digital Models

Analog Models

Algorithms

IMPLEMENTATION

C, C++, VHDL, Verilog, SPICE

MCU, DSP, FPGA, ASIC, Analog Hardware

TEST & VERIFICATION

TEST SYSTEM

MATLAB

Simulink

Stateflow

Fixed-Point Designer

MATLAB Coder & HDL Coder

INTEGRATION
Model-Based Design – HDL & FIL Verification

**RESEARCH**

**REQUIREMENTS**

**DESIGN**
- Environment Models
- Timing and Control Logic
- Digital Models
- Analog Models
- Algorithms

**IMPLEMENTATION**
- C, C++
- VHDL, Verilog
- SPICE
  - MCU
  - DSP
  - FPGA
  - ASIC
  - Analog Hardware

**INTEGRATION**

**TEST & VERIFICATION**
- MATLAB
- Simulink
- Stateflow
- Fixed-Point Designer
- MATLAB Coder & HDL Coder
- HDL Verifier
Model-Based Design – C Code Generation

**RESEARCH**

**REQUIREMENTS**

**DESIGN**
- Environment Models
  - Timing and Control Logic
  - Digital Models
  - Analog Models
- Algorithms

**IMPLEMENTATION**
- C, C++
- VHDL, Verilog
- SPICE
- MCU
- DSP
- FPGA
- ASIC
- Analog Hardware

**TEST & VERIFICATION**

**TEST SYSTEM**

**INTEGRATION**

**MATLAB**

**Simulink**

**Stateflow**

**Fixed-Point Designer**

**MATLAB Coder, Simulink Coder & Embedded Coder**
Questions?
Thank you!